Towards the integration of data-centric distribution technology into partitioned embedded systems

Héctor Pérez
Computers and Real-Time Group
Universidad de Cantabria
Santander, SPAIN
perezh@unican.es

J. Javier Gutiérrez
Computers and Real-Time Group
Universidad de Cantabria
Santander, SPAIN
gutierjj@unican.es

Abstract—**This work proposes an architecture to enable the use of data-centric real-time distribution middleware in partitioned embedded systems based on a hypervisor. Partitioning is a technique that provides strong temporal and spatial isolation, thus allowing mixed-criticality applications to be executed in the same hardware. The proposed architecture not only enables transparent communication among partitions, but it also facilitates the interconnection between partitioned and non-partitioned systems through distribution middleware. Preliminary results show that hypervisor technology provides low overhead and a reasonable trade-off between temporal isolation and performance.**

Keywords—distributed systems; middleware; hypervisor; DDS; real-time systems.

I. Introduction[1]

Partitioning is a widespread technique that enables the execution of multiple applications in the same hardware platform with strong temporal and space isolation, thus allowing the coexistence of mixed-criticality applications, which fulfils their different requirements (i.e. integrity, security, timing, etc.). Although partitioned systems were initially conceived for safety-critical contexts and do not traditionally contemplate the use of distribution middleware because of its complexity, this technique is becoming more and more popular and it is starting to be applied in a heterogeneous set of emerging applications [1].

The use of middleware technology can provide a set of services that may be of interest for partitioned systems, such as location transparency, abstraction of network services, communication management or interoperability. As part of modern model-driven software development techniques, it also may help to resolve key challenges in the development and validation of distributed systems [2] [3]. Over the last years, the Data Distribution Service for Real-Time Systems (DDS) standard [4] has been attracting an increasing interest within the industry due to its flexibility and decoupling capabilities, along with a rich set of *Quality of Service* (QoS) parameters. These features make this standard suitable for the development of distributed systems with real-time requirements [5][6].

1. This work has been funded in part by the Spanish Government and FEDER funds under grant number TIN2011-28567-C03-02 (HIPARTES).

Our concern is to enable partitioned systems to take advantage of common real-time distribution middleware in several scenarios where a high level of criticality is not required. Under these conditions, important design objectives for partitioned systems include software reuse or interoperability between partitioned and non-partitioned systems. Both objectives can be fulfilled by integrating distribution middleware into partitioned systems as shown in [7], which presents an early experience dealing with RT-CORBA [8] and Ada DSA [9] standards. Furthermore, there is an initial attempt to extend DDS with a safety-critical profile [10][11] suitable for partitioned systems such as those defined by ARINC-653 (Avionics Application Standard Software Interface) [12], which proposes this standard as a suitable candidate to interconnect the next-generation of partitioned distributed real-time systems.

Therefore, this paper proposes a system architecture that integrates the use of distribution middleware based on the DDS standard within XtratuM [13], which is an ARINC-653-like hypervisor especially designed for real-time embedded systems. Additionally, a prototype has been developed in order to provide a performance analysis that estimates the overhead incurred when using the proposed architecture. The trade-off between performance and temporal/spatial isolation capabilities is also analysed.

To the best of our knowledge, few research papers have dealt with the merging of DDS and virtualization technology. For instance, the authors in [14] use DDS to interconnect virtual resources on heterogeneous hypervisors. Furthermore, the impact of using DDS in a general-purpose virtualized scenario is addressed in [15]. However, our work differs from these in the target systems, as XtratuM is specially designed to be used in scenarios with hard real-time requirements, in which safety-critical features can be also considered.

This document is organized as follows. Section II introduces the basic characteristics of XtratuM and the DDS standard. The architecture for integrating DDS middleware with XtratuM is proposed in Section III. Section IV describes a potential application as a proof of concept, while Section V evaluates the performance of the proposed architecture. Finally, Section VI draws the conclusions.

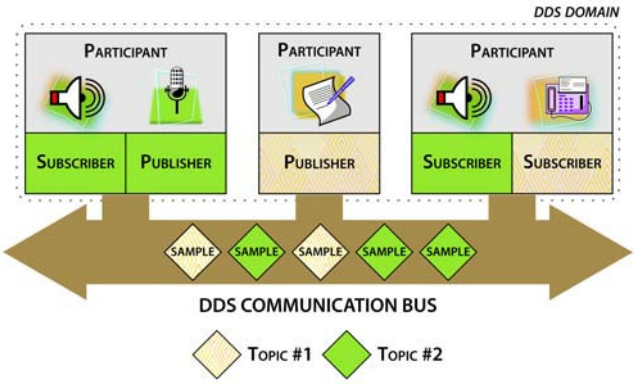

Fig. 1 DDS architecture

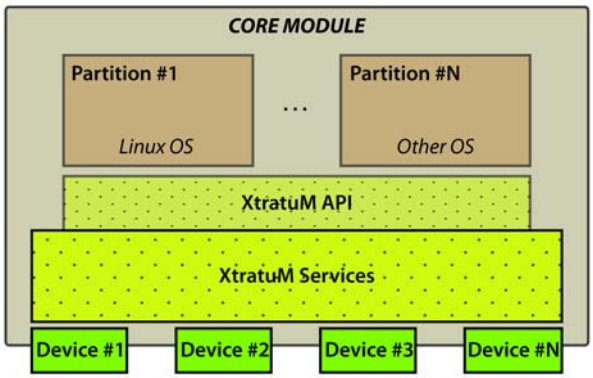

Fig. 2 XtratuM architecture

II. BACKGROUND

A *Overview of DDS*

The Data Distribution Service (DDS) standard defines a data-centric distribution middleware that supports the development of distributed real-time systems [16] by including a wide set of configurable parameters to provide different degrees of QoS. The standard is based on the publisher-subscriber paradigm, where *publishers* and *subscribers* communication entities respectively write (produce) and read (consume) data. All the communication entities that share compatible QoS parameters may be grouped in *participants* of a *domain*, and only entities belonging to the same domain can communicate.

To enable the communication among entities, publishers require to declare their intent to publish a specific *topic* (i.e. the data type to share), while subscribers require to register their interest in receiving particular topics. The example in Figure 1 illustrates a distributed system which consists of three participants in a single domain and two topics. Both topics have a single publisher in charge of generating new data samples. However, successive updates for topic # 1 will only be received by one subscriber, whereas new samples for topic # 2 will be received by two subscribers.

B *Overview of XtratuM*

XtratuM [13] is an open source hypervisor with capabilities to meet real-time and integrity requirements. Although it does not follow a specific standard, its design follows the philosophy of the ARINC-653 avionics standard [12]. This specification defines the interface of a partition-based operating system that allows multiple applications to execute in the same hardware platform, while maintaining time and space isolation. The general architecture of a system using XtratuM is shown in Figure 2, where the term partition represents one or several applications executing over a bare machine or an operating system. Each partition is allocated one or several dedicated time windows during which it may execute and thus multiple partitions can be concurrently executed on the same core module (a hardware platform with one or more processors or cores). Among the facilities provided by XtratuM are the virtualization of the basic resources of the system (clocks, timers, memory, interrupts, etc.) and specific communication services.

Two different and complementary communication services are defined in XtratuM: the ARINC-like communication ports [12] or the XMIO communication service based on Virtio [17]. The former was designed to enable communication in high-integrity systems (e.g., systems with static workload and pre-configured communication links), while the latter is aimed at non-critical software systems with some kind of timing requirements.

In XtratuM, the control and management of devices is left to partitions. To this end, XtratuM provides a configuration service to access the I/O ports, which must be configured at compilation time. I/O ports can belong to only one partition, which means that specific I/O partitions should be created when more than one partition needs to access a particular device. Furthermore, I/O partitions are responsible for implementing the device drivers so devices shared among several partitions should be managed in a special way, as described in the next section.

III. SYSTEM ARCHITECTURE

This section aims to explore the possible architectures that enable the use of data-centric distribution middleware in partitioned systems in which a hypervisor is used to manage the hardware. To guarantee the interoperability among non-critical open subsystems, our proposal will rely on the DDS distribution standard and the XMIO communication service. The analysis for more restrictive scenarios, which may require the use of the ARINC-like communication services and/or a reduced set of the DDS features, is left for future work.

As XtratuM does not implement drivers at the hypervisor level, sharing a device such as a network interface card (NIC) among multiple partitions should focus on handling

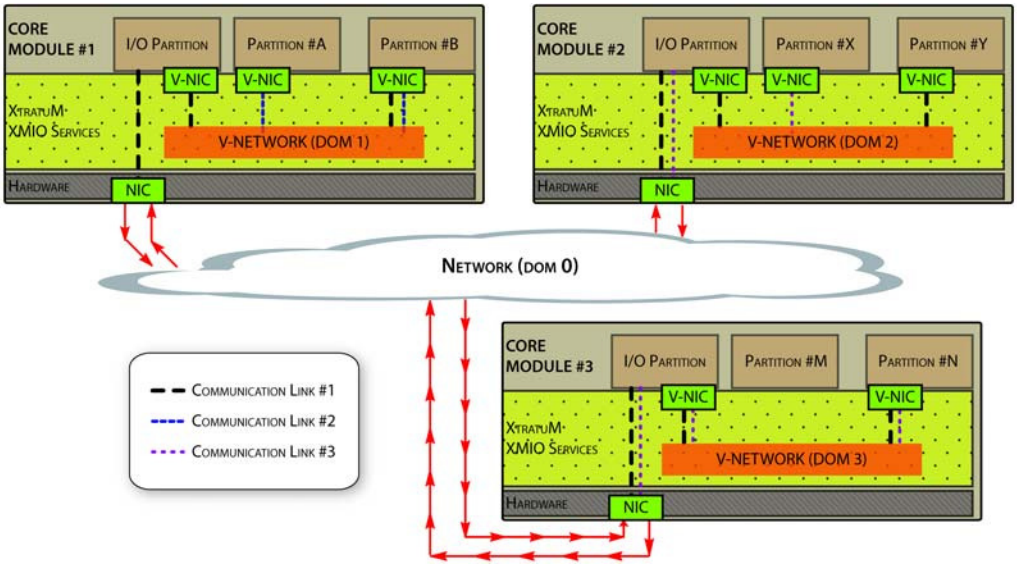

Fig. 3 Proposed architecture for integrating DDS with XtratuM

the contention in order not to compromise both space and time isolation capabilities. A common strategy is the use of an I/O partition that has exclusive access to the network card. Under this approach, the I/O partition is responsible for redirecting messages from the remaining partitions within the same core module to the communications network. To this end, two design strategies could be followed:

- *Designing an I/O partition exclusively aimed at forwarding messages.* In this case, messages are opaque to the I/O partition, and they would be routed through statically established connections. Therefore, each partition should know the destination of each communication link beforehand, which may not be suitable for open systems with variable workload.
- *Considering the use of DDS middleware in the I/O partition.* Thus, data-centric middleware will be responsible for performing routing transparently (e.g., based on topics). In this case, messages are not opaque and can be processed by the I/O partition. Moreover, this option may enable the use of different domains for inter- and intra-communication in core modules, as they may need to maintain certain information contained within.

Hence, each partition should implement data-centric middleware in order to provide distribution facilities such as location transparency, interoperability or connection management, and to facilitate data routing in the case of the I/O partition.

Figure 3 shows a system with three core modules following the proposed architecture for integrating data-centric middleware with a partitioned system using XtratuM. Communications between partitions, belonging or not to the same core module, are performed via DDS. As can be seen in the figure, each core module provides: (1) a virtual network (V-NETWORK) to enable the communication among partitions within the core module, which denotes a DDS domain; (2) a virtual network card (V-NIC) for each partition; and (3) an I/O partition, with exclusive access to the network card, which is responsible for routing the messages received by the underlying communication network, and which is part of another DDS domain. In this case, we have defined three communication links that interconnect partitions: link #1 defines one-to-many communications (i.e., one publisher and several subscribers); link #2 defines one-to-one communications within the same core module; and link #3 defines one-to-one communications between different core modules.

IV. USAGE SCENARIO: VIDEO-SURVEILLANCE SYSTEMS

This section describes a video-surveillance system as a proof of concept in which the use of the proposed architecture can be advantageous. Built-in video-surveillance applications will probably become common in the near future, for example in vehicles for recording unexpected situations (accidents, thefts, etc.). A key feature for this kind of systems resides in the reliability of the recording application, as it must keep recording data continuously, so it can benefit from strong isolation capabilities and can be executed together with other applications. In our example, a distributed application with multiple display monitors may request video captures from the recording application. The architecture for the proposed system is depicted in Figure 4 and it is composed of:

- One core module with two partitions: the *Video_Recorder* partition which is responsible for obtaining data from the attached video cameras and serving the requested video captures to other partitions, and the *Routing_Service* partition which is in charge of

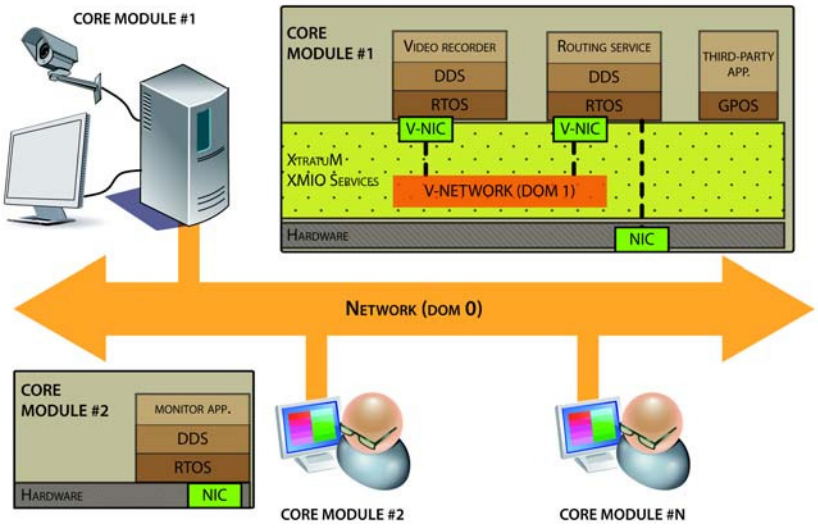

Fig. 4 Scheme of a video-surveillance system

routing the data from/to other core modules spread across the distributed systems.

- A variable number N of core modules that may request the current live video stream or a previous recording (i.e. the monitoring subsystem). These nodes or core module may or may not be partitioned systems.

The use of DDS enables the interoperability among the video recorder and the monitoring subsystems, regardless of whether they are partitioned or not. Furthermore, it also enables the interconnection among the partitions within the same core module (e.g., *Video_Recorder* and *Routing_Service* partitions). The *Routing_Service* partition also relies on DDS to control the information that flows in and out of the core module by providing data distribution between domains. Finally, third-party applications can be easily integrated into the system without compromising the security and data integrity of the *Video_Recorder* partition, as they are isolated in terms of space and time.

V. PERFORMANCE METRICS

This section aims to obtain preliminary performance metrics and assess the interoperability capabilities of using data-centric middleware in partitioned systems by simulating the video-surveillance scenario described in the previous section. In this evaluation, the distributed application consists of two nodes: the video recorder partitioned subsystem and one monitoring non-partitioned subsystem. The hardware platform consists of two single core 2.8 GHz nodes connected through an isolated Gigabit switch in which internal traffic has been disabled (for instance, network packets coming from the Spanning Tree or ARP protocols). We have adapted and integrated in a software platform: RTI Connext DDS[1] as distribution middleware, a fully pre-emptive Linux kernel 2.6.30.5 as the operating system and XtratuM as the hypervisor. Furthermore, a DDS add-on included in the RTI toolsuite[1] has been used to implement the routing service.

In the case of partitioned systems, the optimal configuration of partitions to maximize the processor's utilization is not a trivial problem, and it is even harder with inter-partitions dependencies. Thus, an I/O partition should be executed with sufficient regularity to fulfil the I/O requirements of other partitions. In our example, it is expected that the execution time of middleware operations will be similar to the ones associated with the routing operations, as both rely on DDS middleware. Hence, the video-surveillance application has been configured to have a dedicated time window of 800µs for the *Routing_Service*, and 700µs for the *Video_Recorder* partition, resulting in a scheduling plan repeated every 1,500µs.

The test will measure the execution time of a remote operation that publishes the requested video frames. We measure the operation carried out from the time when the request of a video capture is made until the image is returned. This operation is executed 10,000 times, and the average, maximum, and minimum times are estimated, together with the standard deviation and the 99th percentile (i.e., the value below which 99 percent of the measurements are found). To avoid additional overheads in the measurements, the test is executed without requiring network fragmentation (i.e., the payload is bounded to 1 kilobyte). The performance analysis includes two case-studies.

The first case study, which is called the *overhead test*, aims to estimate the overhead added by XtratuM when it is used as hypervisor. Three scenarios have been defined for this case study:

1. RTI-DDS toolsuite is available at http://www.rti.com

- *Network*, which estimates the temporal cost of using the network (transmitting and receiving a message of 1 kilobyte) by implementing the test over UDP in isolation.
- *Traditional DDS*, which measures the performance of the video-surveillance application using DDS over two non-partitioned nodes.
- *Single DDS Partition*, which measures the performance of the video-surveillance application when the node under analysis is partitioned with XtratuM. In this case, the core module only holds one partition that executes the application and has exclusive access to the network device. Therefore, this scenario estimates the overhead of using XtratuM.

The second case-study (*performance test*) evaluates the performance of the proposed system architecture, that is, with one partition dedicated to the I/O operations. To perform a fairer comparison, this case is contrasted with the traditional distributed application in which a routing service has been added. Therefore, two scenarios have been defined:

- *Traditional DDS with Routing*, which measures the performance of the video-surveillance application using DDS over two non-partitioned nodes. One of the nodes also executes a routing application to enable the communication between domains.
- *Partitioned DDS with Routing,* which measures the performance of the video-surveillance application when the proposed partitioned architecture is applied to one node. In this case, the core module holds two partitions: (1) the *Video_Recorder* partition, and (2) the *Routing_Service* partition to enable the communication between domains.

The results of the analysis for the *overhead test* are shown in Table 1. As can be observed, the DDS example adds a minimum overhead to the network test which makes it suitable for developing our approach, as it requires a lightweight middleware implementation in each partition. Likewise, the maximum overhead of using the distributed application on top of XtratuM is less than 60µs. Taking these metrics into account, it is shown that using hypervisor technology with data-centric middleware is highly efficient.

Table 1: Measurements of response times for the overhead test (in µsecs)

	Min	Avg	Max	Std	Per99
NETWORK	154	206	262	20	249
TRADITIONAL DDS	218	286	415	29	383
SINGLE DDS PARTITION	262	331	467	28	409

Table 2 shows the results of the measurements taken for the *performance test*, in which the proposed system architecture adds complexity by integrating a routing service into the distributed application. As shown in Table 2, the distributed operation for the DDS with routing scenario takes a maximum of 1,632µs, while this value is 4,157µs for the partitioned system. This variation in performance depends on the nature of the partitioned systems and their time window configuration (e.g., a network message received during the execution of the *Video_Recorder* partition has to wait until the next time window corresponding to the *Routing_Service* partition). In our example, we use a time window configuration that allows Linux partitions to be executed properly, as the optimization of time windows for this particular application is beyond the scope of this paper. In any case, the increase in the response times corresponds to a reasonable number of measurements for less critical applications (see the 99th percentile).

Table 2: Measurements of response times for the performance test (in µsecs)

	Min	Avg	Max	Std	Per99
TRADITIONAL DDS WITH ROUTING	662	764	1632	36	876
PARTITIONED DDS WITH ROUTING	1028	1858	4157	539	3346

To complete the study, an additional test has been carried out to evaluate the impact of the proposed architecture with different workloads. Figure 5 depicts the results obtained for the same experiment but using different image sizes. Similarly to the results obtained in Table 1 and Table 2, it is shown that the hypervisor adds a minimum overhead to the traditional DDS scenario regardless of the payload, and the maximum response times are appreciably higher for the partitioned system due to the inherent effect produced by the temporal partitioning. As a consequence of these results, a significant improvement is expected when using a

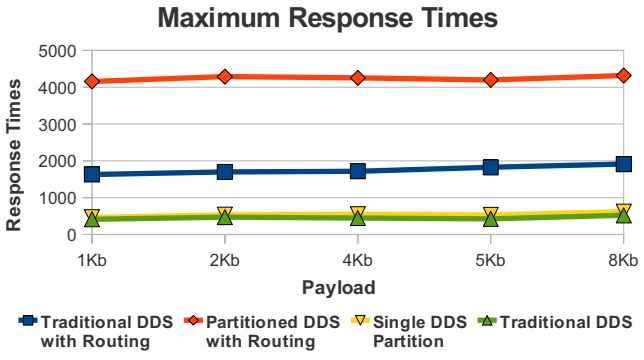

Fig. 5 Maximum response times for different image sizes (in µsecs)

multiprocessor approach that allows, for example, one core to be dedicated to communications, which could avoid the extra delays inherent to the time window configuration. This approach is planned for future work.

VI. Conclusions and Future Work

An increasingly important trend in many domains, such as the automotive, energy distribution or industrial control ones, is support for mixed-criticality applications within the same hardware platform. In this kind of applications, there is also a need to address the integration with the underlying communication subsystem. The proposed integration of DDS data-centric middleware into partitioned systems provides important benefits such as (1) the transparent invocation of services allocated in partitions, independently of whether they are in the same processor (or core) or in different ones; (2) the abstraction of network services which allows the application code to be simplified while maintaining it independent from the communication subsystem; and (3) interoperability between partitioned and non-partitioned systems, or between two or more heterogeneous partitions, e.g., with different levels of criticality or using different data representations (e.g., endianness).

As a consequence of the response times obtained in the performance analysis, it can be observed that the overhead of using data-centric middleware together with a partitioned system could be reasonable for a wide range of applications with soft real-time requirements. However, a significant improvement is expected when using the hypervisor technology adapted to multiprocessor systems, as it may partially mitigate the delays associated with the configuration of time windows. Anyway, it has been shown that this configuration is not a trivial problem and it represents a key step in the design of distributed applications with a partitioned architecture.

Although this integration can facilitate the use of partitioned systems with DDS, further investigation is required to fully determine which features of the standard can be applied, i.e., the applicability of some QoS configurations. Furthermore, it could be interesting to explore other approaches such as the use of the ARINC-like communication services for the incoming safety-critical profile of DDS.

References

[1] Multi-cores Partitioning for Trusted Embedded Systems (MULTIPARTES) European Project, Part of the 7th Framework Programme, http://www.multipartes.eu. 2013.

[2] A. Gokhale, K. Balasubramanian, A.S. Krishna, J. Balasubramanian, G. Edwards, G. Deng, E. Turkay, J. Parsons, and D.C. Schmidt, "Model driven middleware: A new paradigm for developing distributed real-time and embedded systems," Science of Computer Programming 73, pp. 39-58, 2008.

[3] K. An, T. Kuroda, A. Gokhale, S. Tambe, and A. Sorbini, "Model-driven Generative Framework for Automated OMG DDS Performance Testing in the Cloud," 12th International Conference on Generative Programming: Concepts & Experiences (GPCE), Indianapolis (USA), 2013.

[4] Object Management Group. Data Distribution Service for Real-time Systems. OMG Document, v1.2, formal/07-01-01. 2007.

[5] M. Ryll, and S. Ratchev, "Application of the Data Distribution Service for Flexible Manufacturing Automation," International Journal of Aerospace and Mechanical Engineering (2:3), pp. 193-200, 2008.

[6] F. Ben Cheikh, M.A. Mastouri, and S. Hasnaoui, "Implementing a Real-Time Middleware Based on DDS for the Cooperative Vehicle Infrastructure Systems," 6th International Conference on Wireless and Mobile Communications (ICWMC), Valencia, (Spain), 2010.

[7] H. Pérez, and J. J. Gutiérrez, "Experience with the integration of distribution middleware into partitioned systems," Proc. of the 17th International Conference on Reliable Software Technologies, LNCS, vol. 7896. Springer, 1-16. 2013.

[8] Object Management Group. Realtime Corba Specification. OMG Document, v1.2. formal/2005-01-04. 2005.

[9] Ada 2012 Reference Manual. Language and Standard Libraries - Intl. Standard ISO/IEC 8652:2012(E). 2012.

[10] R. Wahlin, and G. Hunt, "Towards a Safety Critical profile for DDS," Real-time and Embedded Systems Workshop, Arlington, VA (USA), 2009.

[11] R. Karoui, and A. Corsaro. "Real time Data Distribution for Airborne Systems," Workshop on Real-time, Embedded and Enterprise-Scale Time-Critical Systems, Washington DC, (USA), 2011.

[12] Airlines Electronic Engineering Committee, Aeronautical Radio INC. "Avionics Application Software Standard Interface". ARINC Specification 653-1. March, 2006.

[13] M. Masmano, I. Ripoll, A. Crespo, and J.J. Metge, "Xtratum a hypervisor for safety critical embedded systems," Proc. of the 11th Real-Time Linux Workshop, Dresden (Germany), 2009.

[14] Y. Cho, J. Choi, and J. Choi, "An integrated management system of virtual resources based on virtualization API and data distribution service," Proc. of the ACM Cloud and Autonomic Computing Conference, New York (USA), 2013.

[15] R Serrano-Torres, M García-Valls, and P Basanta-Val, "Virtualizing DDS middleware: performance challenges and measurements," Proc. of 11th IEEE International Conference on Industrial Informatics, Bonchum (Germany), 2013.

[16] H. Pérez, and J. J. Gutiérrez, "On the schedulability of a data-centric real-time distribution middleware," Computer Standards & Interfaces (34:0), pp. 203-211, 2012.

[17] M. Masmano, S. Peiro, J. Sanchez, J. Simo, and A. Crespo, "IO Virtualisation in a Partitioned System," Proc. of the 6th Embedded Real Time Software and Systems (ERTS^2), Paris (France), 2012.